Device Tree Introduction

Walter Zhou
2015-09-01

My Linux development environment

vi + ctag + cscope

```
$ make ARCH=arm tags
$ make ARCH=arm cscope
```

my .vimrc

```
" vim-scripts repos
Bundle 'vim-misc'
Bundle 'a.vim'
Bundle 'ctags.vim'
Bundle 'linuxsty.vim'
Bundle 'The-NERD-tree'
Bundle 'SuperTab'
Bundle 'grep.vim'
Bundle 'minibufexpl.vim'
Bundle 'Source-Explorer-srcexpl.vim'
Bundle 'taglist.vim'
Bundle 'autoload_cscope.vim'
Bundle 'AutoTag'
Bundle 'cguess'
Bundle 'armasm'
Bundle 'AutoClose'

Bundle 'lua.vim'
Bundle 'octave.vim'
Bundle 'python.vim'
Bundle 'ShowTrailingWhitespace'
Bundle 'FencView.vim'
let g:fencview_autodetect=1
Bundle 'pythoncomplete'
Bundle 'LargeFile'
Bundle 'vis'
Bundle 'bufexplorer.zip'
Bundle 'genutils'
Bundle 'The-NERD-Commenter'
let NERDShutUp=1
map <c-h> ,c<space>

filetype plugin indent on    " required!

set helplang=cn
set fenc=utf-8
set autoindent
......
```

my vi config is in \\10.38.50.90\pub\walter\vim

vi IDE

```
devtree.c (.)
devtree.c (.)                                                                                    237x60
 1 " Press ? for help                  0 [1:setup.c][2:devtree.c]*[7:smp.h]*
 2                                     -MiniBufExplorer-                                    1,26          All
 3 >----------Bookmarks-------
 4                                      1 /*
 5 .. (up a dir)                        2 *  linux/arch/arm/kernel/devtree.c
 6 <me/walterzh/work2/yocto/li          3 *
 7 ▸ .git/                              4 *  Copyright (C) 2009 Canonical Ltd. <jeremy.kerr@canonical.com>
 8 ▸ arch/                              5 *
 9 ▸ block/                             6 * This program is free software; you can redistribute it and/or modify
10 ▸ crypto/                            7 * it under the terms of the GNU General Public License version 2 as
11 ▸ Documentation/                     8 * published by the Free Software Foundation.
12 ▸ drivers/                           9 */
13 ▸ firmware/                         10
14 ▸ fs/                               11 #include <linux/init.h>
15 ▸ include/                          12 #include <linux/export.h>
16 ▸ init/                             13 #include <linux/errno.h>
17 ▸ ipc/                              14 #include <linux/types.h>
18 ▸ kernel/                           15 #include <linux/bootmem.h>
19 ▸ lib/                              16 #include <linux/memblock.h>
20 ▸ mm/                               17 #include <linux/of.h>
21 ▸ net/                              18 #include <linux/of_fdt.h>
22 ▸ samples/                          19 #include <linux/of_irq.h>
23 ▸ scripts/                          20 #include <linux/of_platform.h>
24 ▸ security/                         21 #include <linux/smp.h>
25 ▸ sound/                            22
26 ▸ tools/                            23 #include <asm/cputype.h>
27 ▸ usr/                              24 #include <asm/setup.h>
28 ▸ virt/                             25 #include <asm/page.h>
29   .gitignore                        26 #include <asm/smp_plat.h>
30   .mailmap                          27 #include <asm/mach/arch.h>
31   .Sanitize.patch                   28 #include <asm/mach-types.h>
32   COPYING                           29
33   CREDITS                           30
34   cscope.files                      31 #ifdef CONFIG_SMP
35   cscope.out                        32 extern struct of_cpu_method __cpu_method_of_table[];
36   cscope.out.in                     33
37   cscope.out.po                     34 static const struct of_cpu_method __cpu_method_of_table_sentinel
38   Kbuild                            35         __used __section(__cpu_method_of_table_end);
39   Kconfig                           36
40   MAINTAINERS                       37
41   Makefile                          38 static int __init set_smp_ops_by_method(struct device_node *node)
42   README                            39 {
43   REPORTING-BUGS                    40         const char *method;
44   tags                              41         struct of_cpu_method *m = __cpu_method_of_table;
45   .Source_Explorer.swp              42
~                                      43         if (of_property_read_string(node, "enable-method", &method))
~                                      44                 return 0;
~                                      45
~                                      46         for (; m->method; m++)
~                                      47                 if (!strcmp(m->method, method)) {
~                                      devtree.c                                            32,18         Top
~                                     109 #endif
~                                     110 };
~                                     111
~                                     112 struct of_cpu_method {
~                                     113         const char *method;
~                                     114         struct smp_operations *ops;
~                                     115 };
~                                     116
<ome/walterzh/work2/yocto/linux ~/work2/yocto/linux/arch/arm/include/asm/smp.h                    112,8          91%
--No lines in buffer--
||     arm_add_memory
||     early_mem
||     request_standard_resour
||     customize_machine
||     init_machine_late
||     get_total_mem
||     reserve_crashkernel
||     reserve_crashkernel
||     hyp_mode_check
||     setup_arch
||     topology_init
||     proc_cpu_init
||     c_show
||     c_start
||     c_next
||     c_stop
-  devtree.c (/home/walterzh/w
|-   function
||     set_smp_ops_by_method
||     set_smp_ops_by_method
||     arm_dt_init_cpu_maps
||     arch_match_cpu_phys_id
||     arch_get_next_mach
||     setup_machine_fdt
-  smp.h (/home/walterzh/work2
|-   macro
||     __ASM_ARM_SMP_H
||     raw_smp_processor_id
||     CPU_METHOD_OF_DECLARE
|
|-   struct
||     secondary_data
||     smp_operations
||     of_cpu_method
~
~
~
~
~
~
~
~
~
~
~
~
__Tag_List__    98,1          Bot
```

Device Tree Concept

What is "device tree" ?

a data structure and language for describing hardware

“device tree” Purpose:

make a description of hardware that is readable by an operating system

so that the operating system doesn't need to hard code details of the machine.

Effect

data driven platform setup should result in less code duplication and make it easier to support a wide range of hardware with a single kernel image.

device tree documentation in kernel

1. the specific subsystem's general description

for example:
Documentation/devicetree/bindings/mmc/mmc.txt

2. the specific hardware component's description

for example:
Documentation/devicetree/bindings/mmc/sdhci- pxa.txt

How to describe the hardware configuration in Marvell SDK

walter_marvell_sdk/oem/marvell/marvell_6110_mfp_sdk/hardware/devices/config/rtc_config.c

```
void rtc_get_hw_config( rtc_config_t *buffer )
{
    ASSERT( buffer != NULL );
    // Set RTC register base address
    buffer->rtc_base  = AOAPB_RTC_BASE;
    buffer->rtc_atest = AOAPB_BASE + offsetof(AO_APB_CONFIG_REGS_t, RTC_ATEST);

    // Turn the block on
    AO_APB_CONFIG_REGS_t *regs;
    regs = (AO_APB_CONFIG_REGS_t *)AOAPB_AO_CONFIG_BASE;
    regs->PWCR = AO_APB_CONFIG_PWCR_POR_PDWN_REPLACE_VAL(regs->PWCR, 0);
    cpu_spin_delay( WRITE_DELAY_100 );

#ifdef HAVE_POWER_MGR
    void rtc_register_pwr_event();
    rtc_register_pwr_event();
#endif // ifdef HAVE_POWER_MGR
}
```

The old way in qilin project (kernel 3.2)

*preparation.txt

File Edit Search Options Help

```
/home/walterzh/work/qilin/fpga/kernel/kernel/arch/arm/mach-mmp/qilin_fpga.c

static struct soc_camera_link iclink_ov5640_dvp = {
        .bus_id         = 0,            /* Must match with the camera ID */
        .power          = camera_sensor_power,
        .board_info     = &qilin_i2c_camera[0],
        .i2c_adapter_id = 5,
        .module_name    = "ov5640",
};

static struct resource glan_resources[] = {
    {
        .start  = GLAN_PHYS_BASE,
        .end    = GLAN_PHYS_BASE + (0x2000 - 1),
        .flags  = IORESOURCE_MEM
    },
    {
        .start  = IRQ_MMP3_MMC4,
        .end    = IRQ_MMP3_MMC4,
        .flags  = IORESOURCE_IRQ
    }
};

static struct platform_device glan_device = {
    .name           = "synopGMAC",
    .id             = 0,
    .num_resources  = ARRAY_SIZE(glan_resources),
    .resource       = glan_resources
};

static void __init qilin_fpga_init(void)
{
        mmp3_add_uart(1);

        /* on-chip devices */
        /* TWSI/I2C blocks */
        mmp3_add_twsi(1, NULL, NULL, 0);
        mmp3_add_twsi(2, NULL, NULL, 0);
        mmp3_add_twsi(3, NULL, ARRAY_AND_SIZE(qilin_twsi3_info));
        mmp3_add_twsi(4, NULL, ARRAY_AND_SIZE(qilin_twsi4_info));
        mmp3_add_twsi(5, NULL, ARRAY_AND_SIZE(qilin_twsi1_info));
        mmp3_add_twsi(6, NULL, NULL, 0);

        /* audio sspa support */
        mmp3_add_sspa(1);
        mmp3_add_sspa(2);
        mmp3_add_audiosram(&qilin_audiosram_info);
        ......

8. The old way in 3dp project (3.14, non-device-tree way)

/home/walterzh/work/LSP/3DP_Demo/3dp-emmc-usb-uart-lan-done/linux-mrvl/arch/arm/mach-gogin

DECLARE_GPIO_RESOURCE(A, 0, MV31X0_APB_GPIOA_VIRT_BASE, GPIO_START(GPIOA), GPIO_END(GPIOA));
```

The old way in 3dp project (3.14, non-device-tree way)

*preparation.txt

File Edit Search Options Help

```
8. The old way in 3dp project (3.14, non-device-tree way)

/home/walterzh/work/LSP/3DP_Demo/3dp-emmc-usb-uart-lan-done/linux-mrvl/arch/arm/mach-gogin

DECLARE_GPIO_RESOURCE(A, 0, MV31X0_APB_GPIOA_VIRT_BASE, GPIO_START(GPIOA), GPIO_END(GPIOA));
DECLARE_GPIO_RESOURCE(B, 1, MV31X0_APB_GPIOB_VIRT_BASE, GPIO_START(GPIOB), GPIO_END(GPIOB));
DECLARE_GPIO_RESOURCE(C, 2, MV31X0_APB_GPIOC_VIRT_BASE, GPIO_START(GPIOC), GPIO_END(GPIOC));
DECLARE_GPIO_RESOURCE(D, 3, MV31X0_APB_GPIOD_VIRT_BASE, GPIO_START(GPIOD), GPIO_END(GPIOD));
DECLARE_GPIO_RESOURCE(E, 4, MV31X0_APB_GPIOE_VIRT_BASE, GPIO_START(GPIOE), GPIO_END(GPIOE));
DECLARE_GPIO_RESOURCE(F, 5, MV31X0_APB_GPIOF_VIRT_BASE, GPIO_START(GPIOF), GPIO_END(GPIOF));

// I2C
DECLARE_I2C_RESOURCE(0, MV31X0_I2C0_PHYS_BASE, IRQ_I2C0);
DECLARE_I2C_RESOURCE(1, MV31X0_I2C1_PHYS_BASE, IRQ_I2C1);
#if 0
DECLARE_I2C_RESOURCE(2, MV31X0_I2C2_PHYS_BASE, IRQ_I2C2);
DECLARE_I2C_RESOURCE(3, MV31X0_I2C3_PHYS_BASE, IRQ_I2C3);
#endif

// SERIAL
DECLARE_UART_RESOURCE(0);
DECLARE_UART_RESOURCE(1);
DECLARE_UART_RESOURCE(2);
DECLARE_UART_RESOURCE(3);

// SPI
DECLARE_SPI_RESOURCE(0, MV31X0_SPI0_PHYS_BASE, IRQ_SPI0);
DECLARE_SPI_RESOURCE(1, MV31X0_SPI1_PHYS_BASE, IRQ_SPI1);

static const struct flash_platform_data gogin_spi_slave_data = {
    .type               = "m25p64",
    .nr_parts       = ARRAY_SIZE(gogin_nor_spi_flash),
    .parts          = gogin_nor_spi_flash,
};

static struct spi_board_info gogin_spi_slave_info[] = {
        {
                .modalias       = "m25p80",
                .platform_data  = &gogin_spi_slave_data,
                .irq            = -1,
                .max_speed_hz   = 20000000,
                .bus_num        = 2,
                .chip_select    = 1,
        },
};


static gogin_spi_info_t gogin_bspi_plat_data = {
    .bus_clk    = 200,
    .num_cs     = 2,
};

static struct platform_device gogin_bspi = {
    .name               = "marvell_bspi"
```

The old way in 3dp project (3.14, non-device-tree way)

*preparation.txt

File Edit Search Options Help

```
};

static void __init mv31x0_mach_init (void)
{
    struct clk_lookup *cla;

    platform_device_register(&uart0);
    platform_device_register(&uart1);
    platform_device_register(&uart2);
    platform_device_register(&uart3);

    platform_device_register(&gogin_gpioA);
    platform_device_register(&gogin_gpioB);
    platform_device_register(&gogin_gpioC);
    platform_device_register(&gogin_gpioD);
    platform_device_register(&gogin_gpioE);
    platform_device_register(&gogin_gpioF);

    // set up gpio/multi purpose pins for bypass, pullups, etc
    // do after gpio registration, but before anyone needs bypass
    board_do_mpp_config();

    clr_reg_32bits(0x20000000,(void*)0xf8000200);

    platform_device_register(&gogin_i2c0);
    platform_device_register(&gogin_i2c1);
    cla = clkdev_alloc(&i2c_clk0, NULL, "i2c_designware.0");
    clkdev_add(cla);
    cla = clkdev_alloc(&i2c_clk1, NULL, "i2c_designware.1");
    clkdev_add(cla);

    //platform_device_register(&glan_device);

    //platform_device_register(&usb2D_device);
    //platform_device_register(&usb2H_device);
    //platform_device_register(&usb2O_device);

    spi_register_board_info(gogin_spi_slave_info, ARRAY_SIZE(gogin_spi_slave_info));
    platform_device_register(&gogin_bspi);

    platform_device_register(&gogin_spi0);
    platform_device_register(&gogin_spi1);
    cla = clkdev_alloc(&spi_clk0, NULL, "dw_spi_mmio.0");
    clkdev_add(cla);
    cla = clkdev_alloc(&spi_clk1, NULL, "dw_spi_mmio.1");
    clkdev_add(cla);

#ifdef CONFIG_MMC_SDHCI_GOGIN
    //platform_device_register(&sdmmc0_device);
#endif

#ifdef CONFIG_MTD_NAND_PXA3xx
    platform_device_register(&pxa3xx_device_nand);
    cla = clkdev_alloc(&gogin_nand_clk, NULL, "pxa3xx-nand");
```

machine specific initialization in device tree way

```
static void __init mv31x0_mach_init (void)
{
    struct clk_lookup *cla;

        //board_do_mpp_config();
        clr_reg_32bits(0x20000000,(void*)0xf8000200);

#ifdef CONFIG_USB
    /* davep 25-Jul-2014 ; XXX temp disable until can get the mv31x0_mm.c
     * mappings fixed
     */
    usb_phy_init();
#endif

    of_platform_populate(NULL, of_default_bus_match_table,
                 NULL, NULL);
}
```

dts file

Sample : /home/walterzh/work/LSP/3DP_Demo/3.14/linux-mrvl/arch/arm/boot/dts/mv3dp.dts

make ARCH=arm mv3dp.dtb

you could generate the dtb file according to the dtc command

dtc -I dts -O dtb -b 0 -p 1024 -o mv3dp.dtb mv3dp.dts

dtc --- device tree compiler
dts --- device tree source
dtb --- device tree binary

The dts files is in arch/arm/boot/dts directory

Note: You should understand dtb binary format if you want to debug device tree initialization

config related to device tree (in config-3.18.7-yocto-standard)

```
CONFIG_DTC=y
CONFIG_OF=y
# CONFIG_OF_SELFTEST is not set
CONFIG_OF_FLATTREE=y
CONFIG_OF_EARLY_FLATTREE=y
CONFIG_OF_ADDRESS=y
CONFIG_OF_ADDRESS_PCI=y
CONFIG_OF_IRQ=y
CONFIG_OF_NET=y
CONFIG_OF_MDIO=y
CONFIG_OF_PCI=y
CONFIG_OF_PCI_IRQ=y
CONFIG_OF_MTD=y
CONFIG_OF_RESERVED_MEM=y
CONFIG_OF_GPIO=y
```

device tree framework code

1. drivers/of (of means Open Formware, device tree library)

2. arch/arm/kernel/devicetree.c (device tree initialization)

3. the "of" code related to the specific drivers, for example

video/of_videomode.c
video/of_display_timing.c
watchdog/of_xilinx_wdt.c
tty/serial/of_serial.c
clk/ux500/u8500_of_clk.c
input/touchscreen/of_touchscreen.c
ata/pata_of_platform.c
memory/of_memory.c
memory/of_memory.h
regulator/of_regulator.c
mmc/host/of_mmc_spi.c
iommu/of_iommu.c

.....

How device tree initialization

```
=> tftp 600000 uImage
Speed: 1000, full duplex
Using eTSEC0 device
TFTP from server 192.168.0.103; our IP address is 192.168.0.18
Filename ‘uImage’.
Load address: 0x600000
Loading: ##################################################################
#################################################################
done
Bytes transferred = 1838553 (1c0dd9 hex)
=> tftp c00000 dtbfile
Speed: 1000, full duplex
Using eTSEC0 device
TFTP from server 192.168.0.103; our IP address is 192.168.0.18
Filename ‘dtb’.
Load address: 0xc00000
Loading: ##
done
Bytes transferred = 16384 (4000 hex)
=> bootm 600000 - c00000
```

How device tree initialization (continuted)

```
init/main.c

asmlinkage void __init start_kernel(void)
{
        char * command_line;
        extern const struct kernel_param __start___param[], __stop___param[];

        /*
         * Need to run as early as possible, to initialize the
         * lockdep hash:
         */
        lockdep_init();
        smp_setup_processor_id();
        debug_objects_early_init();

        /*
         * Set up the the initial canary ASAP:
         */
        boot_init_stack_canary();

        cgroup_init_early();

        local_irq_disable();
        early_boot_irqs_disabled = true;

/*
 * Interrupts are still disabled. Do necessary setups, then
 * enable them
 */
        boot_cpu_init();
        page_address_init();
        pr_notice("%s", linux_banner);
        setup_arch(&command_line);              <=== setup_arch
        mm_init_owner(&init_mm, &init_task);
        mm_init_cpumask(&init_mm);

        ......
}
```

How device tree initialization (continuted)

```
arch/arm/kernel/setup.c

void __init setup_arch(char **cmdline_p)
{
        const struct machine_desc *mdesc;

        setup_processor();
        mdesc = setup_machine_fdt(__atags_pointer);     <=== fdt means flattened device tree (dtb format)
        if (!mdesc)
                mdesc = setup_machine_tags(__atags_pointer, __machine_arch_type);
        machine_desc = mdesc;
        machine_name = mdesc->name;

        ......

        paging_init(mdesc);
        request_standard_resources(mdesc);

        if (mdesc->restart)
                arm_pm_restart = mdesc->restart;

        unflatten_device_tree();        <=== we get a tree to describe all hardware configuration

// debugging utility from Walter
//      extern void dump_device_tree(struct device_node *root);
//      dump_device_tree(of_allnodes);

        ......
}
```

Platform Identification

in dts file (mv3dp.dts)

```
        model = "mv3dp";
        compatible = "mrvl,mv31x0";
```

in arch/arm/mach-gogin/mv31x0_arch.c

```
static const char * const mv31x0_compat[] = {
        "mrvl,mv31x0",
        NULL
};


DT_MACHINE_START(MV31X0, "mv31x0")
        .map_io             = gogin_map_io,
        .init_machine       = mv31x0_mach_init,
        .init_early         = mv31x0_init_early,
        .dt_compat          = mv31x0_compat,
MACHINE_END
```

The machine's .dt_compat attribute must mach the dts file!

debug device tree initialization utility function

*preparation.txt

File Edit Search Options Help

```
#ifdef MY_GOGIN_DEBUG
void dump_device_tree(struct device_node *root)
{
    struct device_node *dev_node = root;
        struct property *dev_prop = NULL;
    int i;

    for(; dev_node; dev_node = dev_node->allnext) {
        MY_GOGIN_PRINTK("node: %p", dev_node);

        if(dev_node->parent)
            MY_GOGIN_PRINTK("    parent: %p\n", dev_node->parent);
        else
            MY_GOGIN_PRINTK("    parent: root\n");

        if(dev_node->name)
            MY_GOGIN_PRINTK("name: %s\n", dev_node->name);

        if(dev_node->type)
            MY_GOGIN_PRINTK("type: %s\n", dev_node->type);

        MY_GOGIN_PRINTK("phandle: %08x\n", dev_node->phandle);

        if(dev_node->full_name)
            MY_GOGIN_PRINTK("full_name: %s\n", dev_node->full_name);

        dev_prop = dev_node->properties;
        if(dev_prop) {
            MY_GOGIN_PRINTK("properties:\n");
            for(; dev_prop; dev_prop = dev_prop->next) {
                if(dev_prop->name)
                    MY_GOGIN_PRINTK("    name: %s - %d - %08x\n", dev_prop->name, dev_prop->length, dev_prop->unique_id);
                if(dev_prop->value) {
                    unsigned char *p = (unsigned char *)dev_prop->value;
                    MY_GOGIN_PRINTK("    str value: ");
                    if(isprint(p[0]))
                        MY_GOGIN_PRINTK("%s", p);

                    MY_GOGIN_PRINTK("\n");
                    MY_GOGIN_PRINTK("    hex value: ");

                    for(i = 0; i < dev_prop->length; i++)
                        MY_GOGIN_PRINTK("%02x ", p[i]);

                    MY_GOGIN_PRINTK("\n");
                    MY_GOGIN_PRINTK("    ----------------------------\n");
                }
            }
        } else {
            MY_GOGIN_PRINTK("no properties\n");
        }
        MY_GOGIN_PRINTK("\n");
    }
}
```

Runtime kernel configuration

```
chosen {
/* bootargs = "mem=64M console=ttyS0,115200n8 loglevel=8 debug earlycon=uart8250,mmio,0xf8060000,115200n8"; */
bootargs = "mem=64M console=ttyS0,115200n8 loglevel=8 debug";
};
```

You could transfer kernel parameters in dts file.

Note: u-boot(? version) sometime could modify the kernel parameters in dtb, so please check them in kernel initialization.

Device population

When the driver initialization, it could get its hardware configuration data from runtime device tree node that generated by unflatten_device_tree() by the "of" APIs in drivers/of

irq_of_parse_and_map()
of_iomap()
of_property_read_bool()
of_property_read_u8()
of_property_read_u16()
of_property_read_u32()
of_address_to_resource()
.....

device driver in device tree way, VIC

*preparation.txt

File Edit Search Options Help

```
589 #ifdef CONFIG_OF
590 int __init vic_of_init(struct device_node *node, struct device_node *parent)
591 {
592     void __iomem *regs;
593     u32 interrupt_mask = ~0;
594     u32 wakeup_mask = ~0;
595
596     if (WARN(parent, "non-root VICs are not supported"))
597         return -EINVAL;
598
599     regs = of_iomap(node, 0);
600     if (WARN_ON(!regs))
601         return -EIO;
602
603     of_property_read_u32(node, "valid-mask", &interrupt_mask);
604     of_property_read_u32(node, "valid-wakeup-mask", &wakeup_mask);
605
606     /*
607      * Passing 0 as first IRQ makes the simple domain allocate descriptors
608      */
609     __vic_init(regs, 0, interrupt_mask, wakeup_mask, node);
610
611     dump_vic_device(0);
612
613     return 0;
614 }
615 IRQCHIP_DECLARE(arm_pl190_vic, "arm,pl190-vic", vic_of_init);
616 IRQCHIP_DECLARE(arm_pl192_vic, "arm,pl192-vic", vic_of_init);
617 IRQCHIP_DECLARE(arm_versatile_vic, "arm,versatile-vic", vic_of_init);
618
```

device driver in device tree way, VIC

config in dts file

```
vic: interrupt-controller@fffff000 {
    compatible = "arm,pl190-vic";
    #interrupt-cells = <1>;
    #address-cells = <1>;
    interrupt-controller;
    reg = <0xfffff000 0x1000>;
    valid-mask = <0xffffffff>;
};
```

device driver in device tree way, UART

```
uart@f8060000 {
    compatible = "ns16450";
    reg = <0xf8060000 0x100>;
    reg-shift = <2>;
    clock-frequency = <25000000>;
    reg-offset = <0>;
    reg-io-width = <1>;
    interrupts = <1>;              // hwirq is 1 on vic chip
    interrupt-parent = <&vic>;
};
```

device driver in device tree way, UART (continued)

*preparation.txt

File Edit Search Options Help

```
drivers/tty/serial/of_serial.c

static struct of_device_id of_platform_serial_table[] = {
        { .compatible = "ns8250",    .data = (void *)PORT_8250, },
        { .compatible = "ns16450",   .data = (void *)PORT_16450, },
        { .compatible = "ns16550a",  .data = (void *)PORT_16550A, },
        { .compatible = "ns16550",   .data = (void *)PORT_16550, },
        { .compatible = "ns16750",   .data = (void *)PORT_16750, },
        { .compatible = "ns16850",   .data = (void *)PORT_16850, },
        { .compatible = "nvidia,tegra20-uart", .data = (void *)PORT_TEGRA, },
        { .compatible = "nxp,lpc3220-uart", .data = (void *)PORT_LPC3220, },
        { .compatible = "altr,16550-FIFO32",
                .data = (void *)PORT_ALTR_16550_F32, },
        { .compatible = "altr,16550-FIFO64",
                .data = (void *)PORT_ALTR_16550_F64, },
        { .compatible = "altr,16550-FIFO128",
                .data = (void *)PORT_ALTR_16550_F128, },
#ifdef CONFIG_SERIAL_OF_PLATFORM_NWPSERIAL
        { .compatible = "ibm,qpace-nwp-serial",
                .data = (void *)PORT_NWPSERIAL, },
#endif
        { .type = "serial",          .data = (void *)PORT_UNKNOWN, },
        { /* end of list */ },
};

static struct platform_driver of_platform_serial_driver = {
        .driver = {
                .name = "of_serial",
                .owner = THIS_MODULE,
                .of_match_table = of_platform_serial_table,
        },
        .probe = of_platform_serial_probe,
        .remove = of_platform_serial_remove,
};
```

device driver in device tree way, UART (continued)

*preparation.txt

File Edit Search Options Help

```
static int of_platform_serial_setup(struct platform_device *ofdev,
                        int type, struct uart_port *port,
                        struct of_serial_info *info)
{
        struct resource resource;
        struct device_node *np = ofdev->dev.of_node;
        u32 clk, spd, prop;
        int ret;

        memset(port, 0, sizeof *port);
        if (of_property_read_u32(np, "clock-frequency", &clk)) {

                /* Get clk rate through clk driver if present */
                info->clk = clk_get(&ofdev->dev, NULL);
                if (IS_ERR(info->clk)) {
                        dev_warn(&ofdev->dev,
                                "clk or clock-frequency not defined\n");
                        return PTR_ERR(info->clk);
                }

                clk_prepare_enable(info->clk);
                clk = clk_get_rate(info->clk);
        }
        /* If current-speed was set, then try not to change it. */
        if (of_property_read_u32(np, "current-speed", &spd) == 0)
                port->custom_divisor = clk / (16 * spd);

        ret = of_address_to_resource(np, 0, &resource);
        if (ret) {
                dev_warn(&ofdev->dev, "invalid address\n");
                goto out;
        }

        spin_lock_init(&port->lock);
        port->mapbase = resource.start;

        /* Check for shifted address mapping */
        if (of_property_read_u32(np, "reg-offset", &prop) == 0)
                port->mapbase += prop;

        /* Check for registers offset within the devices address range */
        if (of_property_read_u32(np, "reg-shift", &prop) == 0)
                port->regshift = prop;

        /* Check for fifo size */
        if (of_property_read_u32(np, "fifo-size", &prop) == 0)
                port->fifosize = prop;
```

How device match driver?

How the driver or the device to find its partner ?

The devices of our SoC are all "platform" device, so they are all attached to virtual "platform" bus.

*preparation.txt

File Edit Search Options Help

```
in drivers/base/platform.c (take care of "platform" bus)

static int platform_match(struct device *dev, struct device_driver *drv)
{
        struct platform_device *pdev = to_platform_device(dev);
        struct platform_driver *pdrv = to_platform_driver(drv);

        /* Attempt an OF style match first */
        if (of_driver_match_device(dev, drv))
                return 1;

        /* Then try ACPI style match */
        if (acpi_driver_match_device(dev, drv))
                return 1;

        /* Then try to match against the id table */
        if (pdrv->id_table)
                return platform_match_id(pdrv->id_table, pdev) != NULL;

        /* fall-back to driver name match */
        return (strcmp(pdev->name, drv->name) == 0);
}
```

Trace the matching logic

```
static inline int of_driver_match_device(struct device *dev,
 const struct device_driver *drv)
{
return of_match_device(drv->of_match_table, dev) != NULL;
}
```

drivers/of/device.c

```
const struct of_device_id *of_match_device(const struct of_device_id *matches,
   const struct device *dev)
{
if ((!matches) || (!dev->of_node))
return NULL;
return of_match_node(matches, dev->of_node);
}
```

```
static int __of_device_is_compatible(const struct device_node *device,
    const char *compat, const char *type, const char *name)
{
struct property *prop;
const char *cp;
int index = 0, score = 0;

/* Compatible match has highest priority */
if (compat && compat[0]) {
prop = __of_find_property(device, "compatible", NULL);
for (cp = of_prop_next_string(prop, NULL); cp;
    cp = of_prop_next_string(prop, cp), index++) {
if (of_compat_cmp(cp, compat, strlen(compat)) == 0) {
score = INT_MAX/2 - (index << 2);
break;
}
}
if (!score)
return 0;
}
```

How to locate the driver according to dts

For example:

find the I2C driver in kernel source

```
i2c0@f80a0000 {
compatible = "snps,designware-i2c";
reg = <0xf80a0000 0x1000>;
interrupts = <6>;
interrupt-parent = <&sic0>;
clocks = <&clk_i2c 0>;
};
```

```
linux-mrvl/drivers$ ag "snps,designware-i2c"
i2c/busses/i2c-designware-platdrv.c
300:  { .compatible = "snps,designware-i2c", },
```

The I2C driver source is
i2c/busses/i2c-designware-platdrv.c

dts interpretation - 1

```
vic: interrupt-controller@fffff000 {
compatible = "arm,pl190-vic";
#interrupt-cells = <1>;
#address-cells = <1>;
interrupt-controller;
reg = <0xfffff000 0x1000>;
valid-mask = <0xffffffff>;
};
```

#interrupt-cells = <1>;

define the interrupt format

reg = <0xfffff000 0x1000>;

The IC's address space.

interrupt-controller; ===> means IC

dts interpretation - 2

```
mmc@f9080000 {
    compatible = "mrvl,sdhci-gogin";
    reg = <0xf9080000 0x100>;
    interrupts = <1>;
    interrupt-parent = <&sic0>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_sd0_grp>;
};
```

interrupt-parent = <&sic0>;

means the device is attached to which IC

pinctrl-names = "default";

pinctrl-0 = <&pinctrl_sd0_grp>;

the device use which pins (defined in pin configuration)

pin configuration

The sd0 hardware component use the GPIOB module's pins.

```
mmc@f9080000 {
    compatible = "mrvl,sdhci-gogin";
    reg = <0xf9080000 0x100>;
    interrupts = <1>;                // hwirq is 1 on sic0 chip
    interrupt-parent = <&sic0>;
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_sd0_grp>;
};
```

pinctl configuration (pinctrl driver is responsible for making correct setting)

```
pinctrl: pinctrl@f8040000 {
#address-cells = <1>;
#size-cells = <1>;
compatible = "mrvl,gogin-pinctrl", "simple-bus";

sd0 {
pinctrl_sd0_grp: sd0_grp {
mrvl,pins = <GOGIN_GPIOB 18 GOGIN_ALTFUNC_1    0x00011000
 GOGIN_GPIOB 19 GOGIN_ALTFUNC_1  0x00011000
 GOGIN_GPIOB 20 GOGIN_ALTFUNC_1  0x00011000
 GOGIN_GPIOB 21 GOGIN_ALTFUNC_1  0x00011000
 GOGIN_GPIOB 22 GOGIN_ALTFUNC_1  0x00000000
 GOGIN_GPIOB 23 GOGIN_ALTFUNC_1  0x00011000
 GOGIN_GPIOB 24 GOGIN_ALTFUNC_GPIO  0x00000001
 GOGIN_GPIOB 25 GOGIN_ALTFUNC_GPIO  0x00000000>;
};
};
......
};
```

The Multiplex Pins

1. PINCTRL
2. GPIO
3. Interrupt Controller (IC)

pin configuration (continuted)

*preparation.txt

File Edit Search Options Help

```
static int really_probe(struct device *dev, struct device_driver *drv)
{
        int ret = 0;

        ......

        /* If using pinctrl, bind pins now before probing */
        ret = pinctrl_bind_pins(dev);
        if (ret)
                goto probe_failed;

        if (driver_sysfs_add(dev)) {
                printk(KERN_ERR "%s: driver_sysfs_add(%s) failed\n",
                        __func__, dev_name(dev));
                goto probe_failed;
        }

        if (dev->bus->probe) {
                ret = dev->bus->probe(dev);
                if (ret)
                        goto probe_failed;
        } else if (drv->probe) {
                ret = drv->probe(dev);
                if (ret)
                        goto probe_failed;
        }

        driver_bound(dev);
}
```

*preparation.txt

File Edit Search Options Help

```

static struct platform_driver gogin_rtc_driver = {
    .probe      = gogin_rtc_probe,
    .remove     = __exit_p(gogin_rtc_remove),
    .driver     = {
        .name   = DRV_NAME,
        .owner  = THIS_MODULE,
#ifdef CONFIG_PM
        .pm     = &gogin_rtc_pm_ops,
#endif
    },
};

```

G2 dts files introduction

pegmatile.dtsi
mv6270.dtsi
pegmatite-clocks.dtsi
mv6270-toc-revc.dts
mv6270-ffc_r2-revc.dts

FAQ